

[INICIAR SESIÓN](#)[NUESTROS PLANES](#)[TODOS LOS CURSOS](#)[FORMACIONES](#)[CURSOS](#)[PARA EMPRESAS](#)[ARTÍCULOS DE TECNOLOGÍA > PROGRAMACIÓN](#)

# Redondeo de números en Java



mario-alvial

05/02/2021

En un sistema desarrollado para el área de telemarketing de una empresa, se encontró un error al mostrar el promedio de notas de los asistentes. Tenemos la siguiente tabla de notas para mostrar visualmente el problema:

Empleado: MarceloNota: 7,3213832

Empleado: YuriNota: 6.18317113

Empleado: MárioNota: 9,3181

Ten en cuenta que en esta tabla todos los valores están quebrados, con varios decimales. Sin embargo, para presentar las notas a los empleados, debemos simplificar este valor para que se vea mejor.

Para corregir este error, primero veamos el método que calcula y devuelve el promedio:

```
public static double calculaPromedio(Usuario usuario) {  
  
    double total = 0.0;  
  
    for (Double nota : usuario.getNotas()) {  
  
        total += nota;
```

```
}

return total / usuario.getNotas().size();

}
```

Realmente, el promedio no tiene cuidado con el redondeo. El `double` resultante tendrá el número de cifras decimales que se acerque más al resultado verdadero de la operación matemática, respetando el límite de cifras decimales del tipo primitivo `double`.

El primer paso para solucionar este problema es crear un método que se encargue exclusivamente de redondear el valor calculado por el método `calculaPromedio()`. Entonces, incluso sin haber creado este método, lo devolveremos en `calculaPromedio()`. De esa forma:

```
public static double calculaPromedio(Usuario usuario) {

    double total = 0.0;

    for (Double nota : usuario.getNotas()) {

        total += nota;

    }

    return redondear(total / usuario.getNotas().size());

}
```

## Creando la función de redondeo

Ya di spoiler de cómo será nuestro método, recibe como parámetro un `double` que es el resultado de `calculaPromedio` y devuelve un `double` también, que es el resultado de `calculaPromedio()` redondeado. Así que comencemos a crear este método:

```
private static double redondear(double promedio) {  
}
```

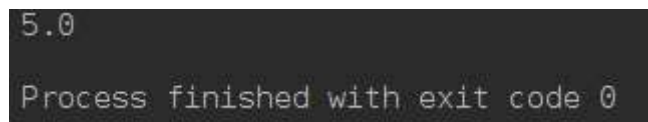
Para redondear nuestro double, podemos usar el método estático [round\(\)](#) de clase [Math](#):

```
private static double redondear(double promedio) {  
    return Math.round(promedio);  
}
```

Tomando como ejemplo un empleado que tenía las siguientes notas: 4,0, 3,7 y 7,8, representados por este código:

```
public static void main(String[] args) {  
    Usuario usuario = new Usuario();  
    usuario.getNotas().add(4.0);  
    usuario.getNotas().add(3.7);  
    usuario.getNotas().add(7.8);  
    System.out.println(calculaPromedio(usuario));  
}
```

Su promedio, en la calculadora, sería: 5,166666667. Veamos el resultado que nos trae nuestro método:

A screenshot of a terminal window with a dark background. The first line shows the output '5.0' in a light blue font. The second line shows 'Process finished with exit code 0' in a light green font.

Ya es una mejora, porque al menos no tenemos esa infinidad de decimales. Pero, el valor real sería 5,166666667 y obtuvimos 5. Entonces no es muy preciso, ¿verdad?

Esto se debe a que el método `round()` devuelve el long más cercano al valor real dado en el parámetro. En nuestro caso, tenemos aproximadamente el valor real de 5,2. Por lo tanto, el long más cercano es 5.

Olvidando el 5 que obtuvimos volveremos a intentar con las mismas notas que usamos en el ejemplo anterior, pero primero mejoraremos nuestro método.

Para mejorar este redondeo, podemos utilizar una solución alternativa. Pensando que el método `round()` redondea a la long más cerca, podemos multiplicar el promedio por 100, use el `round()` y luego dividir por 100. Haciendo esto en el código, tenemos:

```
private static double redondear(double promedio) {  
    return Math.round(promedio * 100.0)/100.0;  
}
```

Ejecutando el código, nuestro resultado es:

```
5.17  
Process finished with exit code 0
```

Entendamos lo que pasó. Primero tomamos nuestro promedio que en nuestro caso tiene un valor de 5,166666667 y lo multiplicamos por 100. Con eso obtuvimos 516,6666667, luego usamos el `round()` que redondeó al long más cercano, 517, y finalmente dividimos 517 entre 100, obteniendo 5,17.

Está bien, ¿verdad? Probemos ahora con otros valores. Las notas del empleado ahora serán: 3,1, 4,3 y 2,5. El resultado teóricamente sería 3,30. Vamos a ver:

```
3.3  
Process finished with exit code 0
```

Date cuenta de que nuestro resultado es correcto, pero solo tiene un decimal. Este es el problema con el método que estamos usando para redondear, no podemos predecir cuántos lugares decimales tendrá el resultado.

¿Qué sería ideal? Idealmente, todos los promedios deberían tener exactamente dos lugares decimales. Para ayudarnos a hacer esto, podemos usar la clase [DecimalFormat](#). Esta clase nos ayuda a formatear números para que se muestren correctamente.

## Redondeo con DecimalFormat

Para empezar, creemos una instancia:

```
DecimalFormat df = new DecimalFormat("0.00");
```

*"0.00" ¿Qué es eso que pones?*

Entendamos. Cada 0 significa que el dígito correspondiente a ese decimal se mostrará con 0 o no. El siguiente paso es pensar en cómo se redondeará nuestro método. Para esto usaremos el método [setRoundingMode\(\)](#):

```
df.setRoundingMode(RoundingMode.HALF_UP);
```

En esta línea estamos diciendo que redondearemos de la mitad a la parte superior, es decir, si el promedio es 3,625, será 3,63.

Por fin, formateemos nuestro promedio usando el método `format()` y ya devuelve este valor. De esa forma:

```
private static String redondear(double promedio) {  
    DecimalFormat df = new DecimalFormat("0.00"); df.setRoundingMode(RoundingMode.  
    return df.format(promedio);  
}
```



Tenga en cuenta que al hacer esto recibimos un error de compilación. Esto sucedió porque nuestro método `redondear()` espera devolver un `double`, pero el método `format()` devuelve un `String`. ¿Y ahora?

Si nos detenemos a pensar, incluso tiene sentido devolver un `String`, porque ya hemos realizado todas las operaciones matemáticas que necesitamos hacer. Ahora solo necesitamos devolver el valor formateado para que lo vea el usuario. Por lo tanto, como ya no necesitaremos operar sobre el promedio, devolver lo mismo como `String` tiene sentido.]

*Pero ¿no podríamos simplemente convertir esta String en un double y devolver este valor?*

Sí, podemos, pero corremos el riesgo de perder el formateo realizado, por lo que optaremos por realmente devolver una `String`. Entonces, cambiemos nuestros métodos para que devuelva `String`.

## Conclusión

Listo, formateamos los datos correctamente. El redondeo a veces es una tarea complicada, dependiendo de la situación necesitamos valores extremadamente precisos, como trabajar con dinero.

Para manejar dinero lo más adecuado es utilizar la clase [BigDecimal que tiene una alta precisión](#). Para comprender mejor cómo es trabajar con redondeo y especialmente cuando se trata de dinero, tenemos el curso de [Java en Alura](#).

Puedes leer también:

- [Intercambiando caracteres de una String en Java](#)
- [Cómo convertir de String para Date en Java](#)
- [Diferencia entre int e Integer en Java](#)

ARTÍCULOS DE TECNOLOGÍA > PROGRAMACIÓN

**En Alura encontrarás variados cursos sobre Programación.  
¡Comienza ahora!**

**SEMESTRAL**

**US\$49,90**

un solo pago de US\$49,90

- ✓ 218 cursos
- ✓ Videos y actividades 100% en Español
- ✓ Certificado de participación
- ✓ Estudia las 24 horas, los 7 días de la semana
- ✓ Foro y comunidad exclusiva para resolver tus dudas
- ✓ Acceso a todo el contenido de la plataforma por 6 meses

**¡QUIERO EMPEZAR A ESTUDIAR!**

[Paga en moneda local en los siguientes países](#)

**ANUAL**

**US\$79,90**

un solo pago de US\$79,90

- ✓ 218 cursos
- ✓ Videos y actividades 100% en Español
- ✓ Certificado de participación
- ✓ Estudia las 24 horas, los 7 días de la semana
- ✓ Foro y comunidad exclusiva para resolver tus dudas
- ✓ Acceso a todo el contenido de la plataforma por 12 meses

**¡QUIERO EMPEZAR A ESTUDIAR!**

[Paga en moneda local en los siguientes países](#)

Acceso a todos  
los cursos

Estudia las 24 horas,  
dónde y cuándo quieras

Nuevos cursos  
cada semana

## NAVEGACIÓN

PLANES

INSTRUCTORES



BLOG

POLÍTICA DE PRIVACIDAD

TÉRMINOS DE USO

SOBRE NOSOTROS

PREGUNTAS FRECUENTES

## ¡CONTÁCTANOS!

¡QUIERO ENTRAR EN CONTACTO!

## BLOG

PROGRAMACIÓN

FRONT END

DATA SCIENCE

INNOVACIÓN Y GESTIÓN

DEVOPS

AOVS Sistemas de Informática S.A

CNPJ 05.555.382/0001-33

## SÍGUENOS EN NUESTRAS REDES SOCIALES



## ALIADOS



En Alura somos unas de las Scale-Ups seleccionadas por Endeavor, programa de aceleración de las empresas que más crecen en el país.



Fuimos unas de las 7 startups seleccionadas por Google For Startups en participar del programa Growth Academy en 2021

POWERED BY

## CURSOS

### Cursos de Programación

Lógica de Programación | Java

### Cursos de Front End

HTML y CSS | JavaScript | React

### Cursos de Data Science

Data Science | Machine Learning | Excel | Base de Datos | Data Visualization | Estadística

### Cursos de DevOps

Docker | Linux

### Cursos de Innovación y Gestión

Productividad y Calidad de Vida | Transformación Ágil | Marketing Analytics |  
Liderazgo y Gestión de Equipos | Startups y Emprendimiento